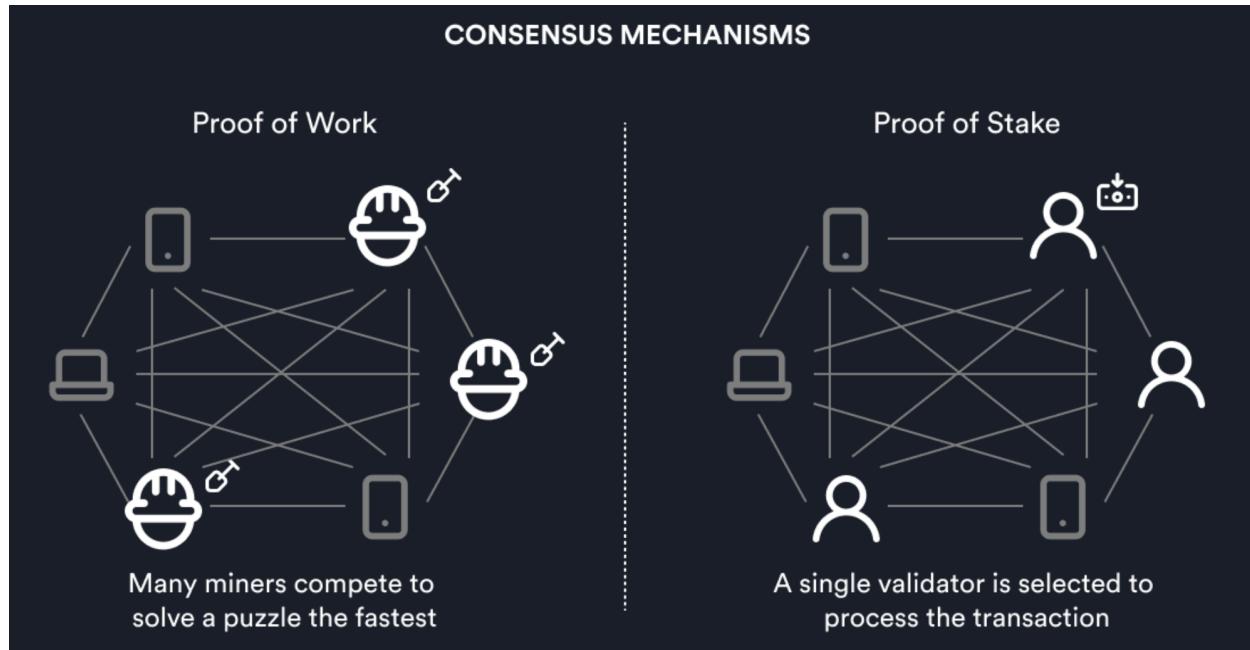


การติดตั้ง Node-jbc บน USB4G Wifi



Validator และ Miners มีบทบาทที่คล้ายคลึงกันในการตรวจสอบข้อมูลและให้การอนุญาตให้ดำเนินการธุรกรรม โดยทั้งคู่มีหน้าที่การยืนยันข้อมูลที่ถูกต้องในเครือข่าย แต่ มีความแตกต่างในการแบ่งส่วนของค่าธรรมเนียมที่ได้รับ ซึ่งขึ้นอยู่กับจำนวนหรือจำนวนที่ถูก Stake ตอนต้น โดยที่ไม่มี การแบ่งขั้นแบบ Proof-of-Work ทำให้กระบวนการนี้มี ลักษณะที่ไม่ต้องใช้พลังงานในการทำงานและเน้นการมี ส่วนร่วมในระบบมากขึ้น.

USB 4G wifi

- SoC – Qualcomm Snapdragon 410 (MSM8916) quad-core Arm Cortex-A53 processor
- System Memory – 512 MB RAM
- Storage – 4GB eMMC flash (3.3 GB available to system)
- Connectivity – 4G LTE modem, WiFi 4
- Debugging – UART



ชุดอุปกรณ์ usb4g ที่ได้ทำการลง os Debian 11 ใหม่ด้วยวิธีการใช้เครื่องมือ platform-tools ของ android

วิธีการติดตั้ง android platform-tools

1 ไปที่ <https://developer.android.com/tools/releases/platform-tools>
เพื่อทำการ download เครื่องมือ

Downloads

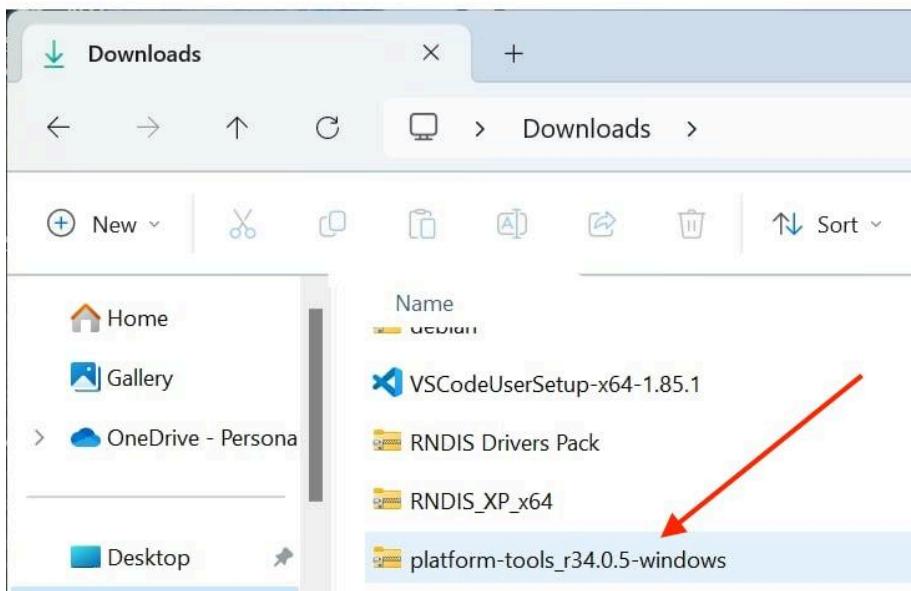
If you're an Android developer, you should get the latest SDK Platform-Tools from Android Studio's [SDK Manager](#) or from the `sdkmanager` command-line tool. This ensures the tools are saved to the right place with the rest of your Android SDK tools and easily updated.

But if you want just these command-line tools, use the following links:

- [Download SDK Platform-Tools for Windows](#)
- [Download SDK Platform-Tools for Mac](#)
- [Download SDK Platform-Tools for Linux](#)

Although these links do not change, they always point to the most recent version of the tools.

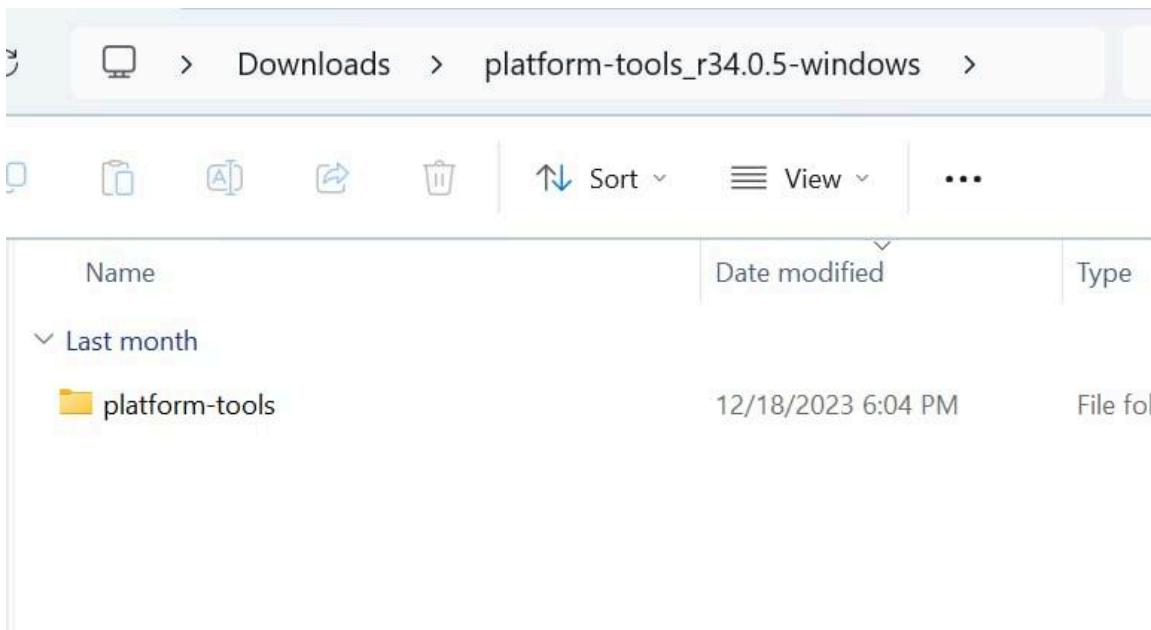
วิธีติดตั้งสำหรับ windows



แตก zip แล้ว copy platform-tools ออกรนาวดใน path ที่เราใช้งาน เช่น วางใน user

C:\user\xxxxx\platform-tools

เวลาเรียกใช้งานใน cmd Command Prompt ตอนไปอยู่ใน path ที่เครื่องมีวางไว้



วิธีติดตั้งสำหรับเครื่อง mac

1 ติดตั้ง Homebrew ผ่าน terminal

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

2 ติดตั้ง platform-tools

```
brew install android-platform-tools
```

ส่วนนี้คือการติดตั้งเครื่องมือบนเครื่องเราเพื่อทำการรีโมทเข้าไปยัง usb 4g ที่ติดตั้งตัว os linux

ขั้นตอนการ shell เข้าไปยังอุปกรณ์ ด้วยคำสั่ง adb shell

MAC เปิด terminal พิม adb shell

```
[tossapornwetsiri@tossaporns-MacBook-Pro ~ % adb shell  
root@openstick:/# ]
```

Windows เข้าถึง platform-tools folder พิม adb shell

```
[ C:\Users\Sense> cd platform-tools
```

```
[root@openstick:/#  
C:\Users\Sense\platform-tools>adb shell
```

**ข้อควรระวังสำหรับการเรียกใช้งาน platform-tools ต้องแน่ใจว่าอยู่ใน folder platform-tools **

ขั้นตอนการเชื่อมต่อ wifi เพื่อต่อ internet

```
[tossapornwetsiri@tossaporns-MacBook-Pro ~ % adb shell  
* daemon not running; starting now at tcp:5037  
* daemon started successfully  
root@openstick:/# ]
```

เมื่อเรา shell สำเร็จแล้ว เราจะรีโมทเข้าไปยัง root@openstick:/# ตามรูป

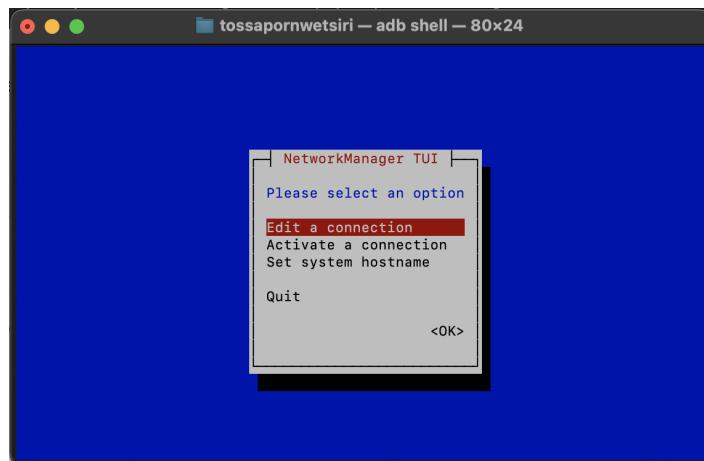
ขั้นตอนต่อไปทำการพิม export TERM=linux และenter

```
[root@openstick:/# export TERM=linux  
root@openstick:/# ]
```

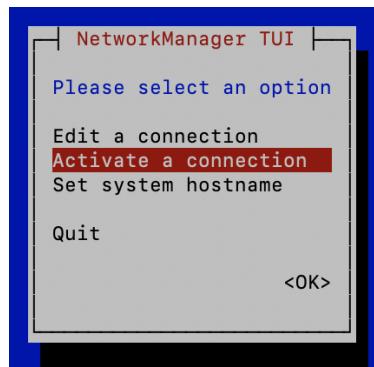
พิม nmtui และenter

```
* daemon not running, starting now at /etc/init.d/NetworkManager
* daemon started successfully
root@openstick:/# export TERM=linux
root@openstick:/# nmtui
```

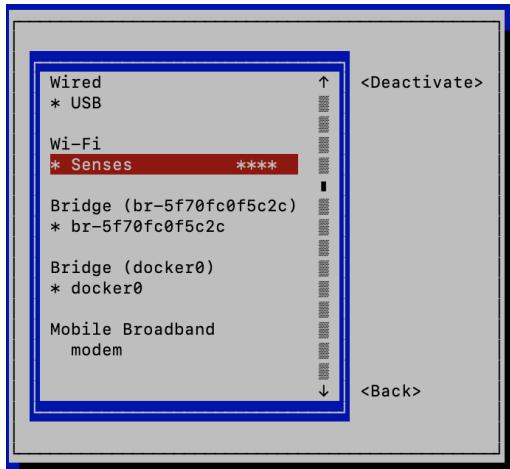
เข้าสู่นี้ สำหรับเชื่อมต่อ wifi



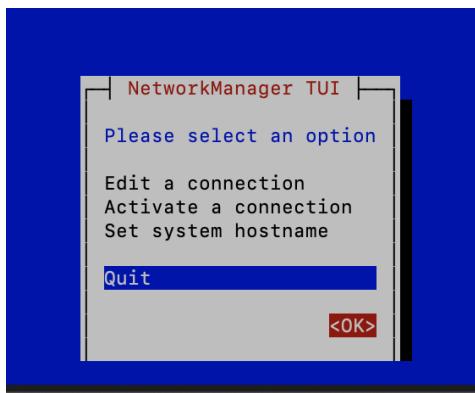
เลือก Activate a connection การเลือกใช้วิธีกดลูกศรขึ้นลง และเลือกกด ok



เลือก ssid ที่เราจะเชื่อมต่อ



ใส่ password ของ ssid ที่เราเลือก



เลิฟแล็ง quit



ติดตั้ง software ที่ต้องใช้ในการสร้าง
deposit_dataxxxxxxx.json และ keystore-m_xxxxxxxxxx.json

sudo apt-get update

root@openstick:/# sudo apt-get update

sudo apt install python3

root@openstick:/# sudo apt install python3

sudo apt install python3-pip

root@openstick:/# sudo apt install python3-pip

sudo apt install git

root@openstick:/# sudo apt install git

ตรวจสอบ python,pip ว่าติดตั้งแล้วหรือยัง

```
[root@openstick:/# python3 --version
Python 3.9.2
[root@openstick:/# pip --version
pip 20.3.4 from /usr/lib/python3/dist-packages/pip (python 3.9)
root@openstick:/# ]
```

ข้อควรจำ ทั้งหมดคือการทำใน shell ที่เป็น linux command ไม่เกี่ยวกับเครื่องที่ใช้ shell เช่นมาดังนั้น
คำสั่งต่างๆคือ linux

ขั้นตอนการ generate validator keys

สามารถดูขั้นตอนได้ใน <https://docs.jibchain.net/nodes-and-validators/generate-validator-keys>

เพื่อใช้ในการอ้างอิงหรือดูรายละเอียดเพิ่ม

1 Clone the Deposit-CLI repository from JIBChain-net Github จาก git ด้วยคำสั่ง

```
git clone https://github.com/jibchain-net/deposit-cli.git && cd deposit-cli
```

2 Install dependencies

```
pip3 install -r requirements.txt
```

```
root@openstick:/# pip3 install -r requirements.txt
```

```
python3 setup.py install
```

```
root@openstick:/# python3 setup.py install
```

หรือใช้ ./deposit.sh install

```
root@openstick:/# ./deposit.sh install
```

3 สร้าง Create key and deposit_data-*.json

สร้างfolder

```
mkdir validator_keys-test
```

```
root@openstick:/deposit-cli# mkdir validator_keys-test
```

3. Create key and deposit_data-*.json

```
mkdir validator_keys-test
```

```
./deposit.sh new-mnemonic --num_validators=<NUM_VALIDATORS> --mnemonic_language=english
```

Example:

```
./deposit.sh new-mnemonic --num_validators=1 --mnemonic_language=english --chain=jib --
```

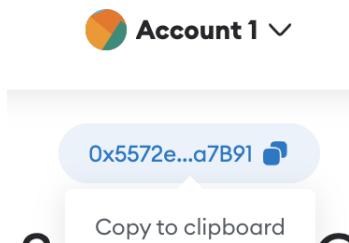
ขั้นตอนนี้เป็นการสร้างตามคำสั่งที่เราจะกำหนดตามตัวอย่าง

```
./deposit.sh new-mnemonic --num_validators=1 --mnemonic_language=english --chain=jib  
--eth1_withdrawal_address=<change-to-your-address> --folder=validator_keys-test
```

แนะนำให้นำคำสั่งไปวางใน textedit ก่อนเพื่อแก้ไข

```
16 January BE 2567 at 00:29  
./deposit.sh new-mnemonic --num_validators=1 --mnemonic_language=english --chain=jib --eth1_withdrawal_address=<change-to-your-  
address> --folder=validator_keys-test|
```

--num_validators=1 คือจำนวน node ที่เราต้องการ Create สามารถปรับได้ตามที่เราต้องการ
--eth1_withdrawal_address=<change-to-your-address> คือ ที่อยู่กระเป่าเราดูได้จาก metamask



เมื่อปรับแล้วให้นำคำสั่งไปวางแล้ว enter

Running deposit-cli...

```
Running deposit-cli...  
***Using the tool on an offline and secure device is highly recommended to keep your mnemonic safe.***  
Please choose your language ['1. Հայերեն', '2. العربية', '3. English', '4. Français', '5. Bahasa melayu',  
'6. Italiano', '7. 日本語', '8. 한국어', '9. Português do Brasil', '10. român', '11. Türkçe', '12. 简体  
中文']: [English]: |
```

เลือกภาษา กดผ่านได้เลย

```
**[Warning] you are setting an Eth1 address as your withdrawal address. Please ensure that you have control over this address.**
```

```
Repeat your execution address for confirmation.: █
```

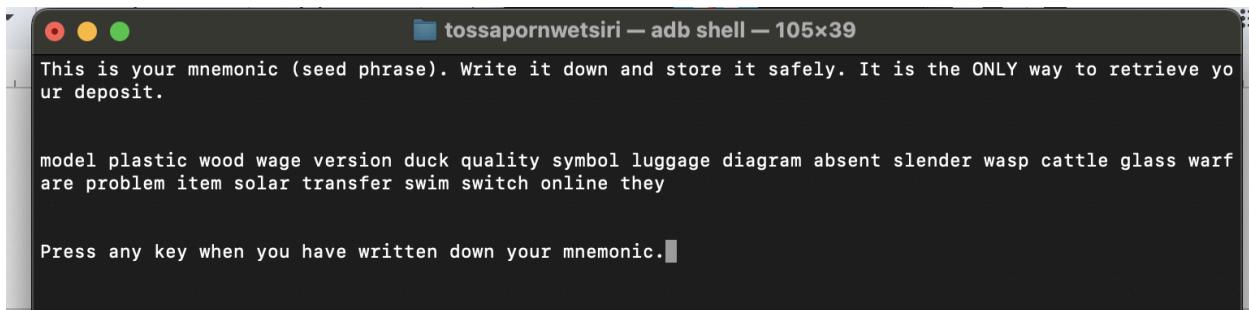
ใส่เลขกระเป้าที่ใช้ในคำสั่งสร้าง key และ deposit_data

```
Create a password that secures your validator keystore(s). You will need to re-enter this to decrypt them when you setup your Ethereum validators.: █
```

ใส่ password 8 ตัว จำให้ดีและตอนใส่ในจ่อจะไม่แสดงอะไร

```
[Create a password that secures your validator keystore(s). when you setup your Ethereum validators.:  
Repeat your keystore password for confirmation: █
```

ใส่ password 8 ตัว confirmation และจำให้ดี ใส่ให้เหมือนครั้งแรก



This is your mnemonic (seed phrase). Write it down and store it safely. It is the ONLY way to retrieve your deposit.

```
model plastic wood wage version duck quality symbol luggage diagram absent slender wasp cattle glass warf are problem item solar transfer swim switch online they
```

Press any key when you have written down your mnemonic. █

Copy seed phrase และเก็บไว้

เอาไปวาง

Please type your mnemonic (separated by spaces) to confirm you have written it down. Note: you only need to enter the first 4 letters of each word if you'd prefer.

2

แล้ว enter

สำเร็จจนขั้นตอนการสร้าง

Deposit_dataxxxxxxxx.json
Keystore-m xxxxxxxxxxx.json

ទរវវនសណុប

cd validator

```
[Press any key.  
root@openstick:/deposit-cli# cd validator_keys-test/validator_keys && ls -la
```

```
keys-test/validator keys && ls -la
```

```
[Press any key.  
[root@openstick:/deposit-cli# cd validator_keys-test/validator_keys && ls -la  
total 56  
drwxrwxrwx 2 root root 4096 Jan 15 17:48 .  
drwxrwxrwx 3 root root 4096 Dec 25 14:51 ..  
-r--r---- 1 root root 702 Dec 25 14:52 deposit_data-1703515927.json  
-r--r---- 1 root root 702 Dec 26 11:02 deposit_data-1703588523.json  
-r--r---- 1 root root 702 Jan 4 16:02 deposit_data-1704384174.json  
-r--r---- 1 root root 702 Jan 13 07:40 deposit_data-1705131633.json  
-r--r---- 1 root root 702 Jan 13 07:54 deposit_data-1705132463.json  
-r--r---- 1 root root 702 Jan 15 17:48 deposit_data-1705340927.json  
-r--r---- 1 root root 710 Dec 25 14:52 keystore-m_12381_3600_0_0_0-1703515925.json  
-r--r---- 1 root root 710 Dec 26 11:02 keystore-m_12381_3600_0_0_0-1703588521.json  
-r--r---- 1 root root 710 Jan 4 16:02 keystore-m_12381_3600_0_0_0-1704384171.json  
-r--r---- 1 root root 710 Jan 13 07:40 keystore-m_12381_3600_0_0_0-1705131631.json  
-r--r---- 1 root root 710 Jan 13 07:54 keystore-m_12381_3600_0_0_0-1705132460.json  
-r--r---- 1 root root 710 Jan 15 17:48 keystore-m_12381_3600_0_0_0-1705340925.json  
root@openstick:/deposit-cli/validator_keys-test/validator_keys# ]
```

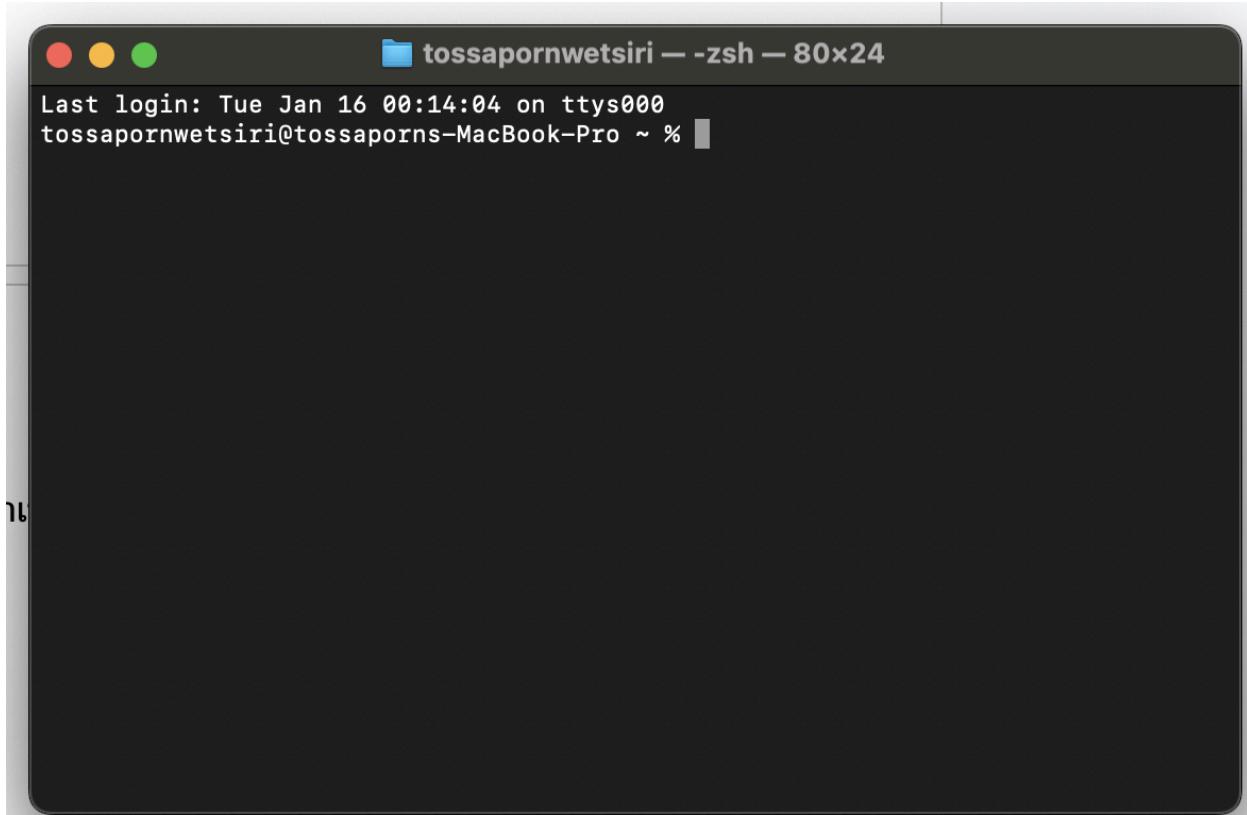
KEYS-TEST/VALIDATOR KEYS XX IS -12

deposit_dataxxxxxx.json

keystore-m_xxxxxxxx.json

```
total 16  
drwxr-xr-x 4 user staff 128 Dec 3 15:14 .  
drwxr-xr-x 3 user staff 96 Dec 3 15:14 ..  
-r--r---- 1 user staff 702 Dec 3 15:14 deposit_data-1701591267.json  
-r--r---- 1 user staff 710 Dec 3 15:14 keystore-m_12381_3600_0_0_0-1701591267.json
```

ดึง deposit_dataxxxxx.json มาที่เครื่องเราเพื่อนำไปStaking



New terminal ขึ้นมาใหม่อีกหนึ่งหน้าต่าง

adb pull /deposit-cli/validator_keys-test/validator_keys/deposit_dataxxxx.json ~/Downloads
สำหรับเครื่องmac เราจะได้ deposit_dataxxxx.json มาอยู่ที่ Downloads

สำหรับ windows

adb pull /deposit-cli/validator_keys-test/validator_keys/deposit_dataxxxx.json
C:\Users\yourname\Downloads

เวลาใช้คำสั่งต้องอยู่ที่หน้า platform-tools
C:\Users\YourUsername\platform-tools:

```
C:\Users\Sense> cd platform-tools
```

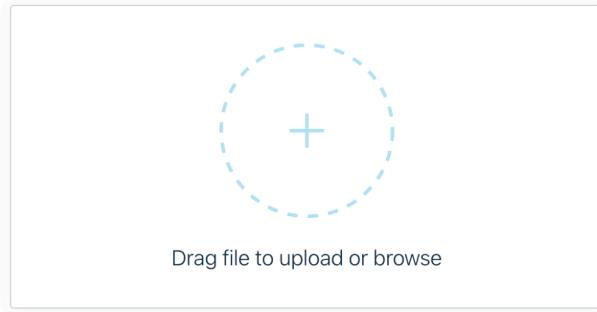
อาจจะมีปัญหาในการดึง deposit_dataxxxx.json ให้สังเกตว่า C:\Users\yourname\Downloads ตรงใหม่ ถ้าใหม่ให้กำหนด path ให้ตรง

นำ deposit_dataxxxx.json ไป staking

<https://staking.jibchain.net/en/>

Upload deposit data

Upload the deposit data file you just generated. The `deposit_data-[timestamp].json` is located in your `/staking-deposit-cli/validator_keys` directory.



วางdeposit_dataxxxxx.json และทำการตามขั้นตอนต่างๆ ในเวปໄได้เลยจนเวปแจ้งการstakingสำเร็จ

Your stake has reached the deposit contract!

You've successfully set up a testnet validator! We recommend you complete the checklist before validating Network

การสร้างและติดตั้งเครื่องมือสำหรับ node-jbc

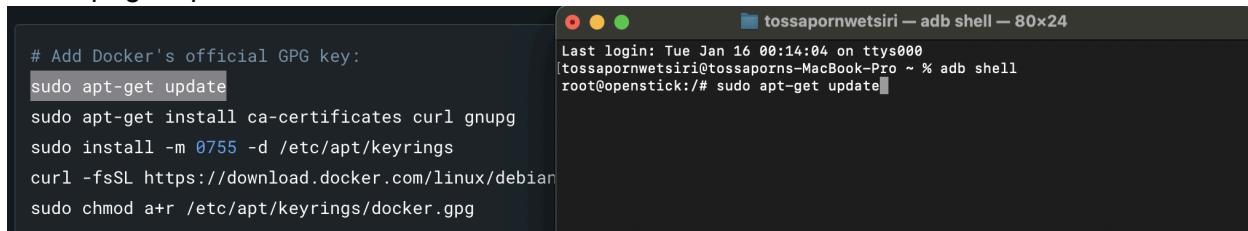
ติดตั้งและสร้าง docker

- Docker
 - Install for Ubuntu : <https://docs.docker.com/engine/install/ubuntu/>
 - Install for Debian : <https://docs.docker.com/engine/install/debian/>

Os ในตัว use 4g เป็น debian ให้คลิกเข้าไปดูได้ที่

<https://docs.docker.com/engine/install/debian/#installation-methods>

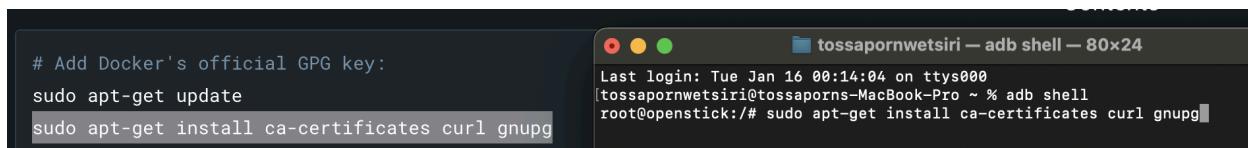
sudo apt-get update



```
# Add Docker's official GPG key:  
sudo apt-get update  
sudo apt-get install ca-certificates curl gnupg  
sudo install -m 0755 -d /etc/apt/keyrings  
curl -fsSL https://download.docker.com/linux/debian  
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

ทำการอัพเดท OS

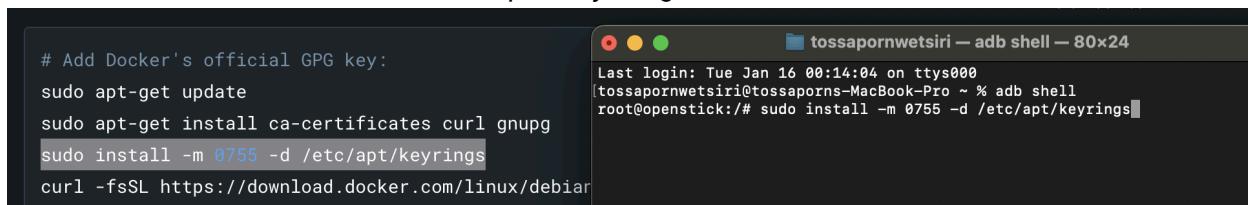
sudo apt-get install ca-certificates curl gnupg



```
# Add Docker's official GPG key:  
sudo apt-get update  
sudo apt-get install ca-certificates curl gnupg
```

ติดตั้ง certificates ของ docker

sudo install -m 0755 -d /etc/apt/keyrings



```
# Add Docker's official GPG key:  
sudo apt-get update  
sudo apt-get install ca-certificates curl gnupg  
sudo install -m 0755 -d /etc/apt/keyrings  
curl -fsSL https://download.docker.com/linux/debian
```

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg
--dearmor -o /etc/apt/keyrings/docker.gpg
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/g
sudo chmod a+r /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/deb
sudo chmod a+r /etc/apt/keyrings/docker.gpg
# Add the repository to Apt sources:
```

Add the repository to Apt sources:
ตรงส่วนนี้ติดตั้ง repository ให้ก็อปทั้งหมด

```
echo \
"deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

```
# Add the repository to Apt sources:
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/ap
$(. /etc/os-release && echo '$VERSION_CODENAME') stable"
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin

ติดตั้ง docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin

```
To install the latest version, run:

$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

ทำการรัน hello-world เพื่อทดสอบ

```
docker run hello-world
root@openstick:/# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

ติดตั้ง nano สำหรับ edit file

sudo apt-get install nano

```
root@openstick:/#
O root@openstick:/# sudo apt-get install nano
```

ขั้นตอนการตรวจสอบเครื่องมือทั้งหมด

Software Checklist

make -v

```
[root@openstick:/# make -v
GNU Make 4.3
Built for aarch64-unknown-linux-gnu
Copyright (C) 1988-2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
root@openstick:/# ]
```

docker -v

```
[root@openstick:/# docker -v
Docker version 24.0.7, build afdd53b
root@openstick:/# ]
```

docker compose version

```
[root@openstick:/# docker compose version
Docker Compose version v2.21.0
root@openstick:/# ]
```

Running a Validator

<https://docs.jibchain.net/nodes-and-validators/become-a-validator/running-a-validator>
สามารถเข้าไปดูจากลิ้งค์ด้านทางเพื่อเป็นเอกสารอ้างอิงได้

Clone JBC Node on your node machine:

```
cd ~  
git clone https://github.com/jibchain-net/node.git jbc-node  
cd jbc-node
```

Check node file ตรวจสอบ node file ที่เราได้ clone มา

```
ls -la
```

Example:

```
total 64  
drwxr-xr-x 14 user  staff  448 Dec  3 16:37 .  
drwxr-xr-x  4 user  staff  128 Dec  3 16:05 ..  
-rw-r--r--  1 user  staff  682 Dec  3 16:07 .env.example  
drwxr-xr-x 14 user  staff  448 Dec  3 16:20 .git  
-rw-r--r--  1 user  staff    9 Dec  3 16:08 .gitignore  
-rw-r--r--  1 user  staff  581 Dec  3 16:17 Makefile  
-rw-r--r--  1 user  staff  569 Dec  3 16:05 README.md  
drwxr-xr-x 11 user  staff  352 Dec  3 16:05 config  
-rw-r--r--  1 user  staff 1518 Dec  3 16:05 docker-compose.yaml  
-rw-r--r--  1 user  staff 1121 Dec  3 16:05 genesis.json  
-rwxr-xr-x  1 user  staff  216 Dec  3 16:05 init-script.sh  
drwxr-xr-x  4 user  staff  128 Dec  3 16:11 keys  
-rw-r--r--  1 user  staff  466 Dec  3 16:19 validator.yaml
```

ทำการสร้าง file env ด้วยครับสั่ง make env

```
root@openstick:/root/jbc-node# make env
```

ทำการแก้ไข file .env ด้วยโปรแกรม nano

sudo nano .env

```
[root@openstick:/root# cd jbc-node
root@openstick:/root/jbc-node# nano .env]
```

```
GNU nano 5.4          .env *
## BOOTNODE Configuration
NETWORK_ID=8899
TARGET_PEERS=100
NODE_PUBLIC_IP=27.145.4.72

## VALIDATOR Configuration
NODE_GRAFFITI=JBCValidatorClient
PUBLIC_BEACON_NODE="https://metrabyte-cl.jibchain.net/"
FEE_RECIPIENT=xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

แก้ไขเลขกระเบ้าเราลงไปที่ FEE_RECIPIENT=xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
เปลี่ยน xxxxxxxxxxxxxxxx เป็นเลขกระเบ้า

กด control x และกดเลือก Y เพื่อทำการ save โดยให้ทันชื่อเดิม

Copy `keystore-m_<timestamp>.json` to folder `keys`

Copy keys

```
cp /deposit-cli/validator_keys-test/validator_keys/keystore-m_xxxxxxxxxxx.json /jbc-node/keys/
```

```
total 16
drwxr-xr-x  4 user  staff  128 Dec  3 15:14 .
drwxr-xr-x  3 user  staff   96 Dec  3 15:14 ..
-r--r----  1 user  staff  702 Dec  3 15:14 deposit_data-1701591267.json
-r--r----  1 user  staff  710 Dec  3 15:14 keystore-m_12381_3600_0_0-1701591267.json
```

```
cp /deposit-cli/validator_keys-test/validator_keys/keystore-m_xxxxxxxxxxx.json
/jbc-node/keys/
```

**** ถ้าไม่ผ่านให้ตรวจสอบมีการติด folder ชื่อ root หรือใหม่ ถ้ามีอาจจะต้องเพิ่มเป็น

```
cp /deposit-cli/validator_keys-test/validator_keys/keystore-m_xxxxxxxxxxx.json
/root/jbc-node/keys/
```

make import-validator-key

```
root@openstick:/root/jbc-node# make import-validator-key
```

6. Import validator key

```
make import-validator-key
```

Example:

```
docker run -ti --rm -v ./data/lighthouse:/root/lighthouse -v ./data/lighthouse/custom/validators:/root/.lighthouse/custom/validators
Running account manager for custom (/config) network
validator-dir path: "/root/.lighthouse/custom/validators"
WARNING: DO NOT USE THE ORIGINAL KEYSTORES TO VALIDATE WITH ANOTHER CLIENT, OR YOU WILL
LOSE YOUR VALIDATOR.

Keystore found at "/keys/keystore-m_12381_3600_0_0_0-1701591267.json":
- Public key: 0x92d768771c062137e7433fd27072330b14b75d94379432098916ea145d648d2aa84daa4
- UUID: 96bcdd0c-e0ef-48c2-8bff-7930965c86aa

If you enter the password it will be stored as plain-text in validator_definitions.yml:
Enter the keystore password, or press enter to omit it:
```

ใช้ keystore password ที่เราตั้งไว้ตอนที่เราสร้าง deposit_data และ keystore-m

```
Create a password that secures your validator keystore(s). You will need to re-enter this to decrypt them
when you setup your Ethereum validators.:
```

ใส่ password 8 ตัว จำให้ดีและตอนใส่ในจอยจะไม่แสดงอะไร

```
[Create a password that secures your validator keystore(s).
when you setup your Ethereum validators.:
Repeat your keystore password for confirmation: ]
```

ใส่ password 8 ตัว confirmation และจำให้ดี ใส่ให้เหมือนครั้งแรก

Password is correct.

Successfully imported keystore.

Successfully updated validator_definitions.yml.

Successfully imported 1 validators (0 skipped).

sudo make run-validator

```
[root@openstick:/root/jbc-node# sudo make run-validator
docker compose -f validator.yaml up -d
[+] Running 2/2
  ✓ Network jbc-node_default  Created0.2s
  ✓ Container jbc-validator    Started0.3s
root@openstick:/root/jbc-node# ]
```

sudo make validator-logs

```
[root@openstick:/root/jbc-node# sudo make validator-logs]
```

เพื่อดูว่าทำงานไหม

```
jbc-validator | Jan 16 04:51:43.995 INFO Validator exists in beacon chain      fee_recipient: 0x5572...  
7b91, validator_index: 14159, pubkey: 0xb74f01cd525aabaa24b405422d756a48b1f4918f8ec3123fedcfb9d81531ca3  
a43a621306b615bac6911a61c48784cc, service: duties  
jbc-validator | Jan 16 04:51:53.001 INFO Connected to beacon node(s)          synced: 1, available:  
1, total: 1, service: notifier  
jbc-validator | Jan 16 04:51:53.002 INFO All validators active                  slot: 351838, epoch: 1  
9994, totalValidators: 1, activeValidators: 1, currentEpochProposers: 0, service: notifier  
jbc-validator | Jan 16 04:52:05.001 INFO Connected to beacon node(s)          synced: 1, available:  
1, total: 1, service: notifier  
jbc-validator | Jan 16 04:52:05.004 INFO All validators active                  slot: 351839, epoch: 1  
9994, totalValidators: 1, activeValidators: 1, currentEpochProposers: 0, service: notifier  
jbc-validator | Jan 16 04:52:17.001 INFO Connected to beacon node(s)          synced: 1, available:  
1, total: 1, service: notifier  
jbc-validator | Jan 16 04:52:17.002 INFO All validators active                  slot: 351840, epoch: 1  
9995, totalValidators: 1, activeValidators: 1, currentEpochProposers: 0, service: notifier  
jbc-validator | Jan 16 04:52:29.001 INFO Connected to beacon node(s)          synced: 1, available:  
1, total: 1, service: notifier  
jbc-validator | Jan 16 04:52:29.002 INFO All validators active                  slot: 351841, epoch: 1  
9995, totalValidators: 1, activeValidators: 1, currentEpochProposers: 0, service: notifier  
jbc-validator | Jan 16 04:52:41.001 INFO Connected to beacon node(s)          synced: 1, available:  
1, total: 1, service: notifier  
jbc-validator | Jan 16 04:52:41.002 INFO All validators active                  slot: 351842, epoch: 1  
9995, totalValidators: 1, activeValidators: 1, currentEpochProposers: 0, service: notifier  
root@openstick:/root/jbc-node#
```

sudo make stop-validator หยุดการทำงาน เพื่อ restart หรืออื่นๆ

หลังติดตั้งและรันแล้วให้เพิ่มคำสั่ง

sudo systemctl enable docker

sudo docker run --restart always -d sigp/lighthouse

เพื่อให้ docker ทำงานอัตโนมัติและ restart lighthouse ให้เรา

หมายเหตุ

*** การ restart node ***

make stop-validator

```
[root@openstick:/jbc-node# make stop-validator
docker compose -f validator.yaml down
[+] Running 2/2
  ✓ Container jbc-validator    Removed0.0s
  ✓ Network jbc-node_default  Removed0.2s
root@openstick:/jbc-node# ]
```

make run-validator

```
[root@openstick:/jbc-node# make run-validator
docker compose -f validator.yaml up -d
[+] Running 2/2
  ✓ Network jbc-node_default  Created0.2s
  ✓ Container jbc-validator    Started0.2s
root@openstick:/jbc-node# ]
```

หลังติดตั้งและรันแล้วให้เพิ่มคำสั่ง

sudo systemctl enable docker

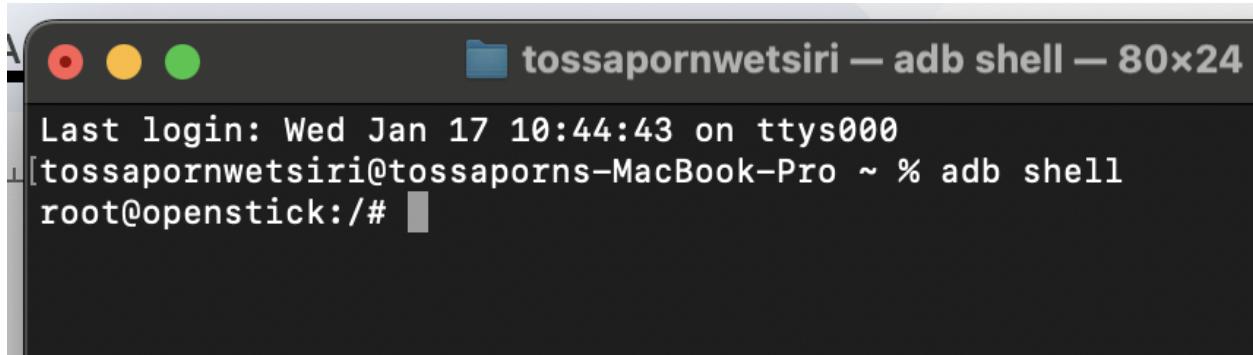
sudo docker run --restart always -d sigp/lighthouse

เพื่อให้ docker ทำงานอัตโนมัติและ restart lighthouse ให้เรา

ภาคผนวก A

การทำ ssh จากรุ่น network เดียวกัน

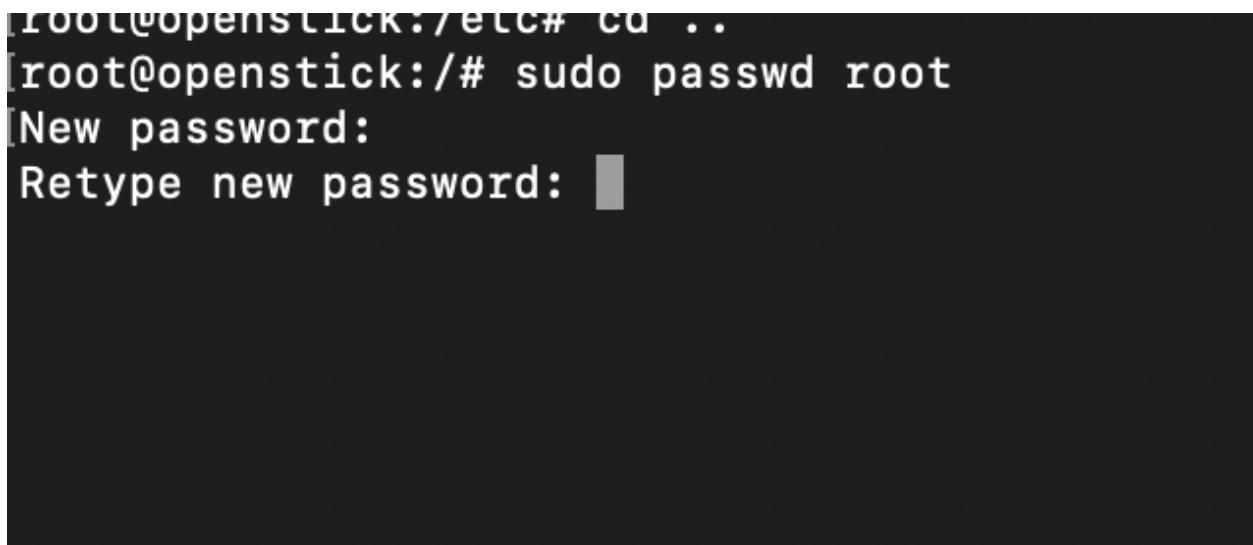
วิธีทำให้เข้า shell ที่ตัวอุปกรณ์ด้วย adb shell ก่อนเพื่อเข้าไปตั้งค่า



```
Last login: Wed Jan 17 10:44:43 on ttys000
[tossapornwetsiri@tossaporns-MacBook-Pro ~ % adb shell
root@openstick:/#
```

sudo passwd root

เพื่อตั้งค่า passในการเข้าทาง ssh



```
[root@openstick:/etc# cd ..
[root@openstick:/# sudo passwd root
[New password:
Retype new password:
```

เข้าไปแก้ไข sshd_config

```
cd etc  
cd ssh
```

```
[root@openstick:/# cd etc  
[root@openstick:/etc# cd ssh  
root@openstick:/etc/ssh# ]
```

```
nano sshd_config
```

```
root@openstick:/etc/ssh# nano sshd_config ]
```

ไปที่ หานบรรทัดที่ มีค่าว่า #PermitRootLogin prohibit-password
แก้ #PermitRootLogin prohibit-password เป็น PermitRootLogin yes
***อย่าลืมเอา # ออกด้วย

```
#SyslogFacility AUTH  
#LogLevel INFO  
  
# Authentication:  
  
#LoginGraceTime 2m  
PermitRootLogin yes  
#StrictModes yes  
#MaxAuthTries 6  
#MaxSessions 10
```

EXIT และ save

ค้นหาเลข ip ด้วยคำสั่ง ifconfig

```
[root@openstick:/# ifconfig  
br-cd134017c4bb: flags=4099<UP,BROADCAST
```

ใส่ดูบอร์ด wlan0:

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      inet 192.168.1.61 netmask 255.255.255.0 broadcast 192.168.1.255  
      inet6 fe80::e3ec:546f:79b9:4d99 prefixlen 64 scopeid 0x20<link>  
      inet6 2001:fb1:62:8c7e:7b56:3fff:8fb8:c07c prefixlen 64 scopeid 0x0<gl
```

เลข IP คือ 192.168.1.61

แล้วเอาอุปกรณ์ไปต่อ กับอุปกรณ์จ่ายไฟ 5 v ทิ่วนะ



เข้า ssh ด้วยคอมพิวเตอร์ที่เชื่อมต่อว่างแลนเดียวกัน

ssh root@192.168.1.61

```
aporns-MacBook-Pro ~ % ssh root@192.168.1.61
```

ใส่ password ที่เราตั้งไว้ก็จะสามารถรีโมทจากวงแลนได้ และเข้าไปทำการ restart node ได้ และตรวจสอบ log ได้โดยไม่ต้องเอามาต่อที่คอมพิวเตอร์

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the
permitted by applicable law.
Last login: Wed Jan 17 03:49:14 2024 from 192.168.1.36
root@openstick:~#
```